EclairJS

Node.js/JavaScript for Spark

EclairJS = Node.js + Apache Spark

Why does this equation make sense?

Node.js is considered best for fast, scalable network applications handling large numbers of simultaneous connections

- Non-blocking event loop

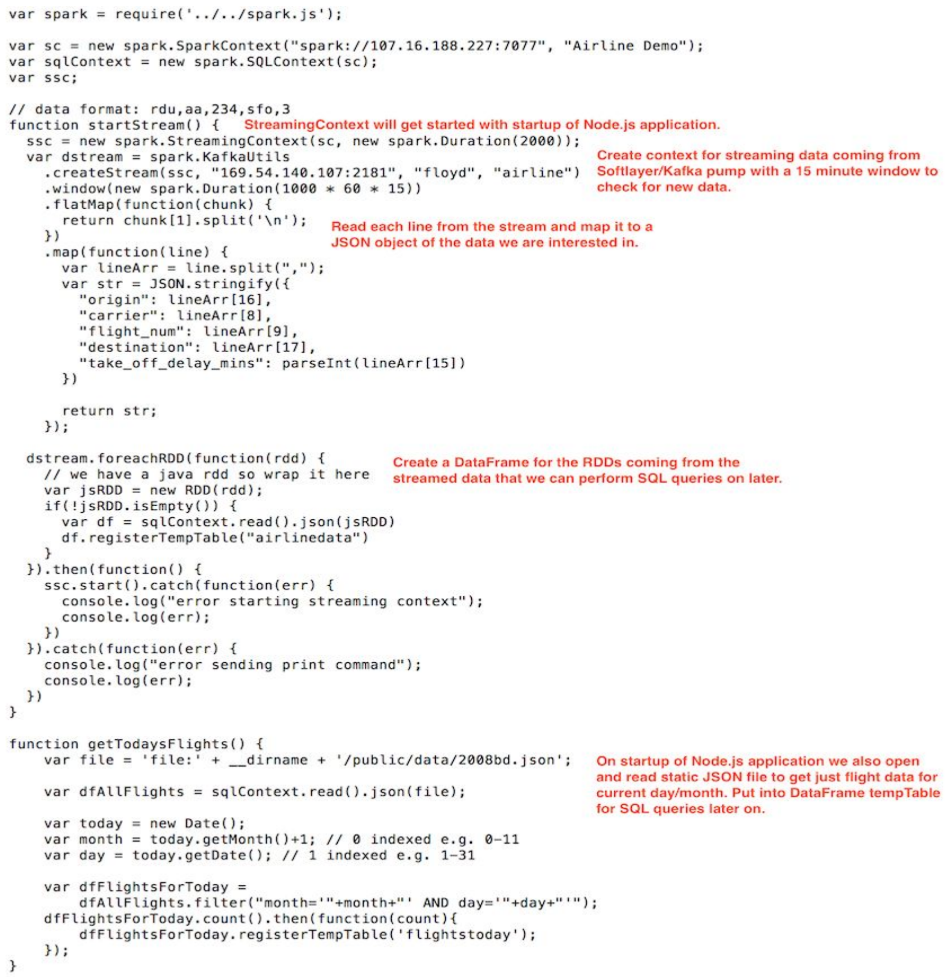
However, “Node.js was never created to solve the compute scaling problem. It was created to solve the I/O scaling problem, which it does really well.” (Tomislav Capan) …

Apache Spark is considered best for fast, scalable compute-intensive applications handling big data, and seamlessly integrating multiple capabilities: streaming, SQL, ML, and graph.

EclairJS enables Node.js developers to program “as-usual” but directly against Spark, taking advantage of its big-data computing and multiple capabilities

Streaming Air Travel Demonstration

Code Example - Main Spark Code



```
var spark = require('../../spark.js');

var sc = new spark.SparkContext("spark://107.16.188.227:7077", "Airline Demo");
var sqlContext = new spark.SQLContext(sc);
var ssc;

// data format: rdu,aa,234,sfo,3
function startStream() {
  ssc = new spark.StreamingContext(sc, new spark.Duration(2000));
  var dstream = spark.KafkaUtils
    .createStream(ssc, "169.54.140.107:2181", "floyd", "airline")
    .window(new spark.Duration(1000 * 60 * 15))
    .flatMap(function(chunk) {
      return chunk[1].split('\n');
    })
    .map(function(line) {
      var lineArr = line.split(",");
      var str = JSON.stringify({
        "origin": lineArr[16],
        "carrier": lineArr[8],
        "flight_num": lineArr[9],
        "destination": lineArr[17],
        "take_off_delay_mins": parseInt(lineArr[15])
      })

      return str;
    });

  dstream.foreachRDD(function(rdd) {
    // we have a java rdd so wrap it here
    var jsRDD = new RDD(rdd);
    if(!jsRDD.isEmpty()) {
      var df = sqlContext.read().json(jsRDD)
      df.registerTempTable("airlinedata")
    }
  }).then(function() {
    ssc.start().catch(function(err) {
      console.log("error starting streaming context");
      console.log(err);
    })
  }).catch(function(err) {
    console.log("error sending print command");
    console.log(err);
  })
}
```

StreamingContext will get started with startup of Node.js application.

Create context for streaming data coming from Softlayer/Kafka pump with a 15 minute window to check for new data.

Read each line from the stream and map it to a JSON object of the data we are interested in.

Create a DataFrame for the RDDs coming from the streamed data that we can perform SQL queries on later.

```
function getTodaysFlights() {
    var file = 'file:' + __dirname + '/public/data/2008bd.json';

    var dfAllFlights = sqlContext.read().json(file);

    var today = new Date();
    var month = today.getMonth()+1; // 0 indexed e.g. 0-11
    var day = today.getDate(); // 1 indexed e.g. 1-31

    var dfFlightsForToday =
        dfAllFlights.filter("month='"+month+"' AND day='"+day+"'");
    dfFlightsForToday.count().then(function(count){
        dfFlightsForToday.registerTempTable('flightstoday');
    });
}
```

On startup of Node.js application we also open and read static JSON file to get just flight data for current day/month. Put into DataFrame tempTable for SQL queries later on.

Code Example - Main Node.js Code

```
var express = require('express');
var app = express();
```

Create a Node.js Express application.

```
app.use(express.static('public'));
app.use(express.static(__dirname + '/public'));

var airlineDemo = require('./airline.js');
```

Get an instance for the class that wraps the main Spark code.

```
app.get('/getFlights', function (request, response) {
  var airportCode = request.query.airport;
  // data format: rdu,aa,234,sfo,3
  try {
    var df = airlineDemo.query("SELECT * FROM airlinedata WHERE origin='"+airportCode+"'");
  } catch (e) {
    console.log("e", e)
  }

  df.toJSON().toArray().then(function(result) {
    response.json(result);
  }).catch(function(e) {
    console.log(e)
  })
});
```

Handle GET requests for streaming flight data; query the DataFrame where we're dumping flights that have departed within last 15 minutes.

```
app.get('/getCarriers', function (request, response) {
    var airportCode = request.query.airport;
    try {
    var carriers = airlineDemo.query("SELECT DISTINCT carrier FROM flightstoday WHERE origin='"+airportCode+"'");
    carriers.cache().toJSON().toArray().then(function(result){
        response.json(result);
    });
  } catch (e) {
    console.log("e", e)
  }
});
```

Handle GET requests to get carriers for a certain airport; queries the DataFrame with static data for current day's flights.

```
app.get('/getSchedule', function (request, response) {
    var airportCode = request.query.airport;
    var carrier = request.query.carrier;
    try {
    var flightsToday = airlineDemo.query("SELECT flight_num,destination FROM flightstoday WHERE origin='" +
        airportCode + "' AND carrier='" + carrier + "'");
    flightsToday.cache().toJSON().toArray().then(function(result){
        response.json(result);
    });
  } catch (e) {
    console.log("e", e)
  }
});
```

Handle GET requests to get flight schedule for a given airline carrier at a given airport; queries the DataFrame with static data for current day's flights.

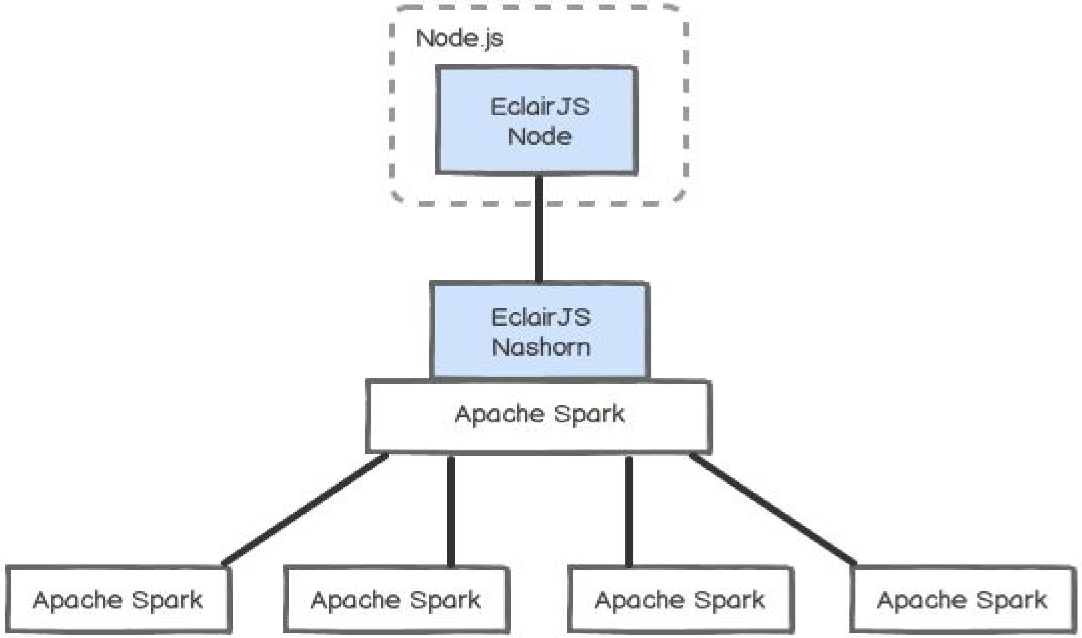
```
var server = app.listen(3000, 'localhost', function () {
    console.log('listening on *:3000');
});
```

Startup the Node.js application.

```
// start the demo
airlineDemo.start();
```

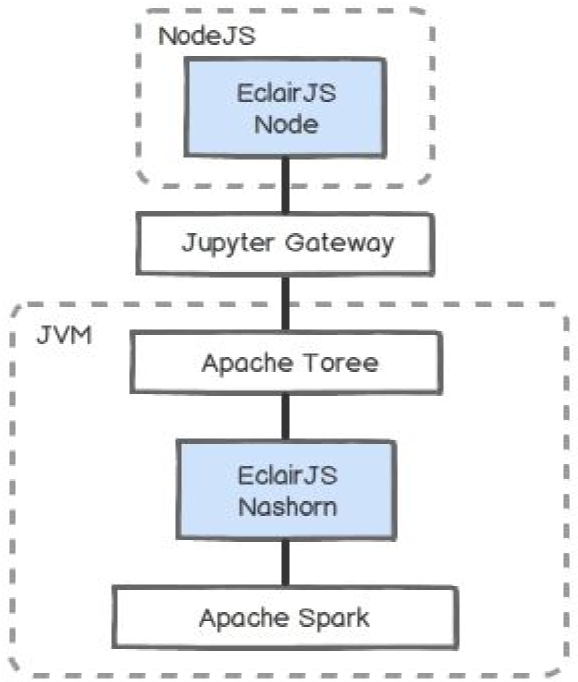
Tell the class that wraps the Spark code to start the stream and start reading the static data.

EclairJS Node & Nashorn



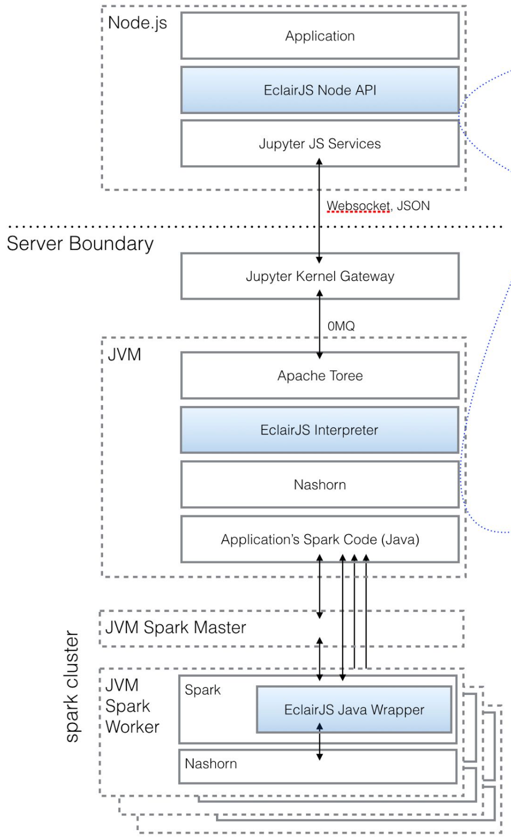
EclairJS Node

EclairJS Node provides Node.js applications with a Spark API through an npm installable client so that Node.js applications can run remotely from the Spark engine.



Apache Toree (Incubator) FKA Spark Kernel

In-Depth Operation



```
var numAs = logData.filter(function(s){
   var ret = (s.indexOf("a") > -1);
   return ret;
});
```

SparkJS Node API & Jupyter JS Services resolve variables & stringify code

```
{code: "var rdd2051768 = rdd4563412.filter(function(s){
  var ret = (s.indexOf("a") > -1);
  return ret;"
}
```

Jupyter components & Apache Toree handle Spark Context, code-snippet evaluation & returning results

SparkJS Interpreter instantiates Nashorn, resolves Spark Java method references & initiates code evaluation

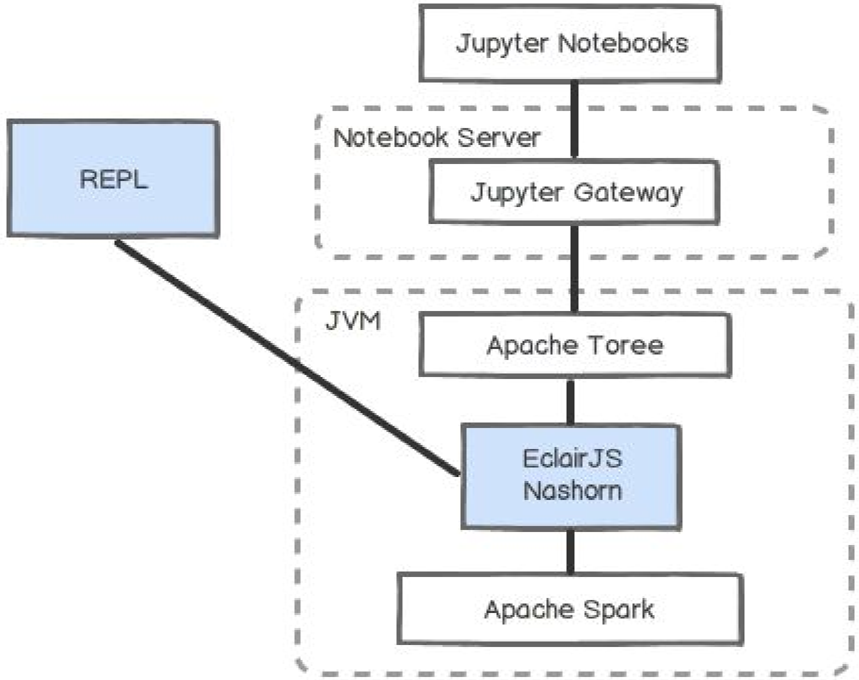
```
return this.jvmRdd.filter(
   new com.ibm.spark.javascript
          .JSFunction("function(s){return (s.indexOf('a') > -1;}")
);
```

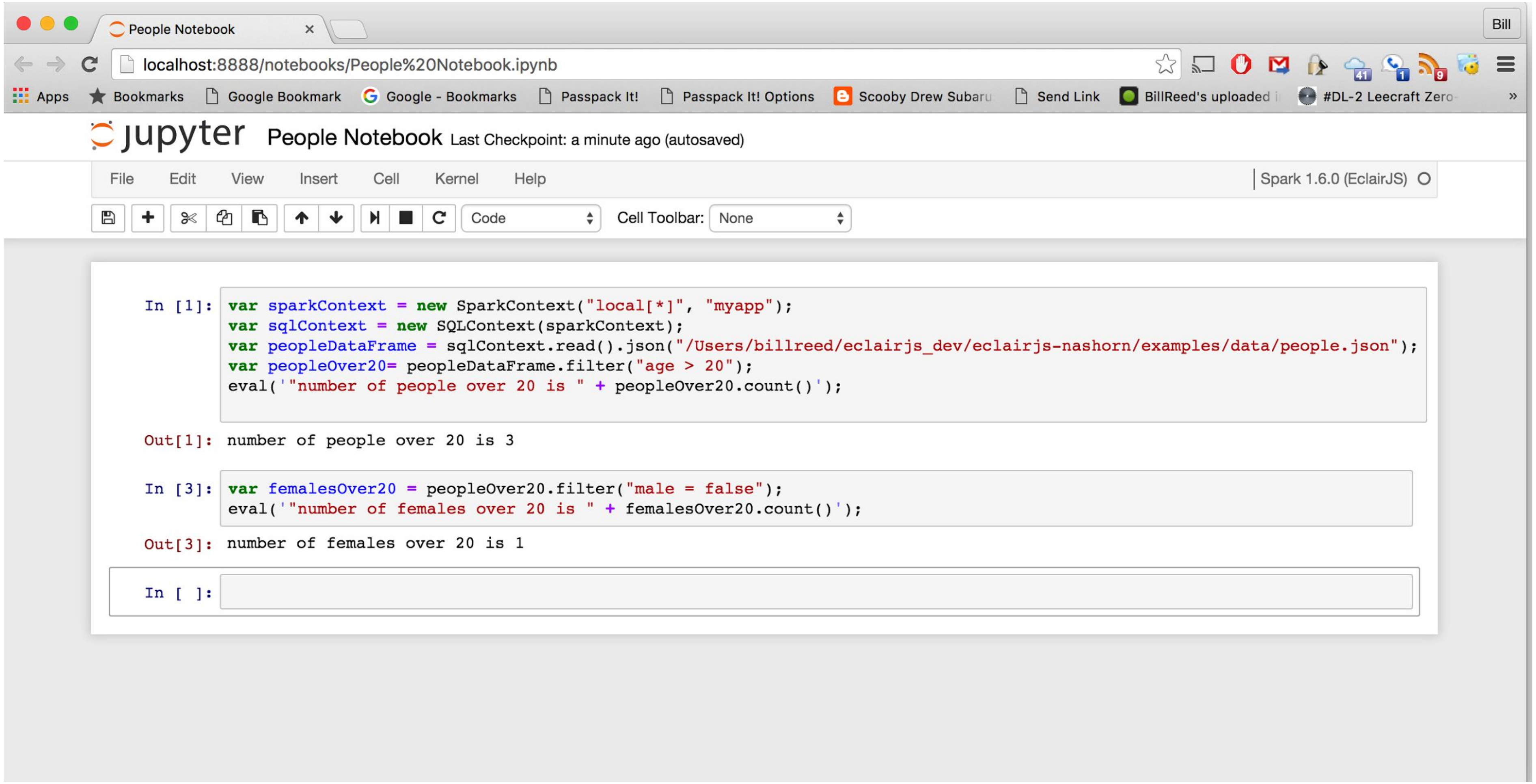
SparkJS Java Wrapper & Nashorn execute lambda functions & handle type conversions on Worker nodes

EclairJS Components

EclairJS Nashorn

EclairJS Nashorn implements the support for JavaScript in Spark, and provides a framework that supports various applications including a REPL and Jupyter Notebooks (and EclairJS Node).





```
In [1]: var sparkContext = new SparkContext("local[*]", "myapp");
        var sqlContext = new SQLContext(sparkContext);
        var peopleDataFrame = sqlContext.read().json("/Users/billreed/eclairjs_dev/eclairjs-nashorn/examples/data/people.json");
        var peopleOver20= peopleDataFrame.filter("age > 20");
        eval('"number of people over 20 is " + peopleOver20.count()');

Out[1]: number of people over 20 is 3

In [3]: var femalesOver20 = peopleOver20.filter("male = false");
        eval('"number of females over 20 is " + femalesOver20.count()');

Out[3]: number of females over 20 is 1

In [ ]:
```

Development

Our application target is the cloud.

Our development cycle is:

- Develop Node.js applications locally with local Spark cluster.
- Optional: Test applications against Spark Cluster (with EclairJS Nashorn etc) in SoftLayer
- Push applications to Bluemix cloud where Node.js is available as a service. Provide SoftLayer Spark cluster reference.

Roadmap

- We are in the process of developing/implementing the entire Spark API.
  - Both **EclairJS** projects are managed in parallel because to develop the *Node* need to develop similar capabilities in the *Nashorn* API, thus the roadmap bel components of EclairJS.
- Milestone 1 API support for:
  - Core
  - Streaming
  - SQL/Dataframe
  - Spark 1.6
- Milestone 2 API support for:
  - MLLib
  - GraphX

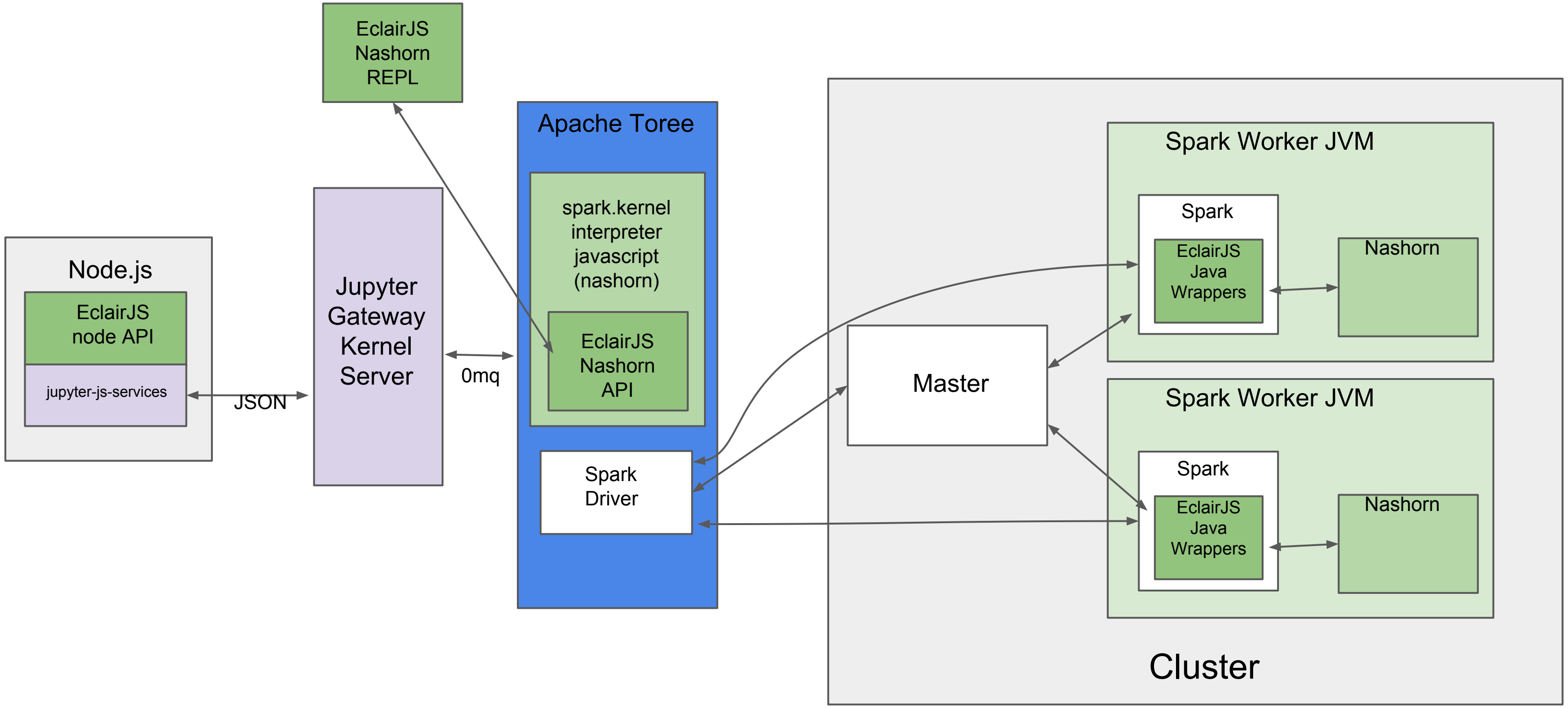
Try It Out & Get Involved!

- JavaScript-enabled Jupyter notebooks running in the IBM Bluemix Cloud.
- Build components from source and setup your own local or clustered environment.
  - Docker Container provides Notebook setup
- All details are documented on GitHub
- Projects are open source on GitHub, Apache v2 license

- The projects are looking for contributors and contributions!
- Google Group - https://groups.google.com/forum/#!forum/eclairjs
- GitHub Project - https://github.com/EclairJS/eclairjs-node/, https://github.com/EclairJS/eclairjs-nashorn

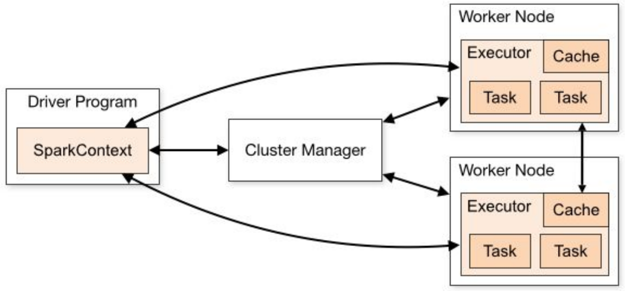
Back up slides

EclairJS Node & Nashorn Detail



Apache Spark

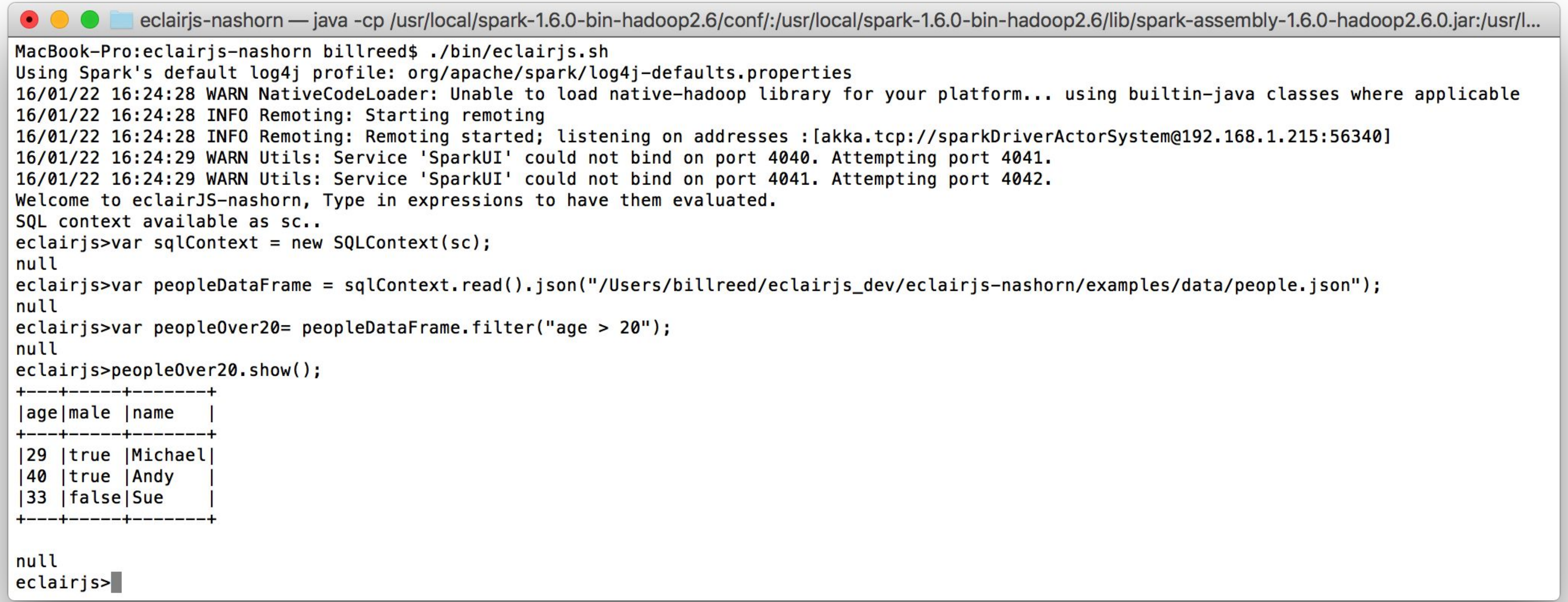
- Fast and general-purpose engine for large-scale data processing
  - Like Hadoop, but using in-memory processing
  - Runs with any Hadoop data source
- Includes libraries for SQL, streaming, machine learning and graph data
  - Can combine multiple libraries within a single application
- Applications can be written in Scala, Java, Python and R
- Scale achieved through parallel distributed-processing: master and workers
- Rapidly being adopted by data engineers for its advanced analytics capabilities, performance, ease of programming
  - Most active Apache project
  - Apache 2 license
  - spark.apache.org



Jupyter

- **Jupyter Notebook** is a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.
- **Jupyter Kernel Gateway** is a Jupyter application that implements different APIs and protocols for accessing Jupyter kernels such as:
  - Accessing HTTP and Websocket resources of the /api/kernels using jupyter/n
    jupyter/jupyter_client and jupyter/jupyter_core
  - Accessing notebook cells via HTTP endpoints

EclairJS REPL Example

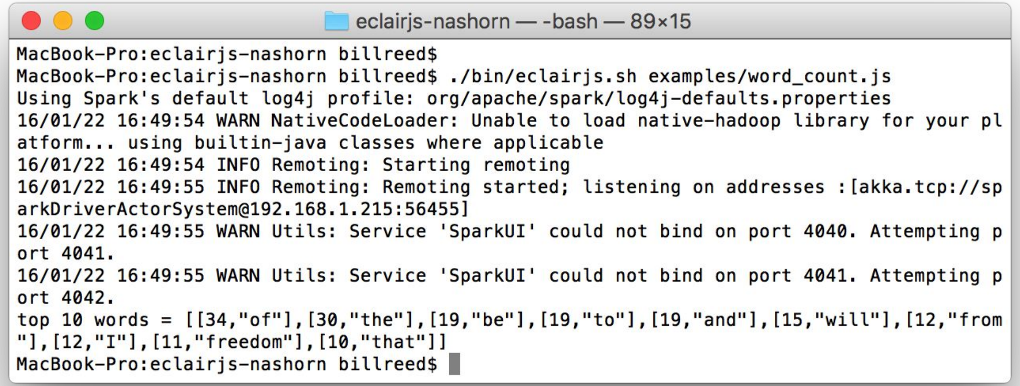


eclairjs-nashorn — java -cp /usr/local/spark-1.6.0-bin-hadoop2.6/conf/:/usr/local/spark-1.6.0-bin-hadoop2.6/lib/spark-assembly-1.6.0-hadoop2.6.0.jar:/usr/l...

```
MacBook-Pro:eclairjs-nashorn billreed$ ./bin/eclairjs.sh
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
16/01/22 16:24:28 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
16/01/22 16:24:28 INFO Remoting: Starting remoting
16/01/22 16:24:28 INFO Remoting: Remoting started; listening on addresses :[akka.tcp://sparkDriverActorSystem@192.168.1.215:56340]
16/01/22 16:24:29 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
16/01/22 16:24:29 WARN Utils: Service 'SparkUI' could not bind on port 4041. Attempting port 4042.
Welcome to eclairJS-nashorn, Type in expressions to have them evaluated.
SQL context available as sc..
eclairjs>var sqlContext = new SQLContext(sc);
null
eclairjs>var peopleDataFrame = sqlContext.read().json("/Users/billreed/eclairjs_dev/eclairjs-nashorn/examples/data/people.json");
null
eclairjs>var peopleOver20= peopleDataFrame.filter("age > 20");
null
eclairjs>peopleOver20.show();
+---+-----+-------+
|age|male |name   |
+---+-----+-------+
|29 |true |Michael|
|40 |true |Andy   |
|33 |false|Sue    |
+---+-----+-------+

null
eclairjs>
```

EclairJS JavaScript Driver Example



```
MacBook-Pro:eclairjs-nashorn billreed$
MacBook-Pro:eclairjs-nashorn billreed$ ./bin/eclairjs.sh examples/word_count.js
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
16/01/22 16:49:54 WARN NativeCodeLoader: Unable to load native-hadoop library for your pl
atform... using builtin-java classes where applicable
16/01/22 16:49:54 INFO Remoting: Starting remoting
16/01/22 16:49:55 INFO Remoting: Remoting started; listening on addresses :[akka.tcp://sp
arkDriverActorSystem@192.168.1.215:56455]
16/01/22 16:49:55 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting p
ort 4041.
16/01/22 16:49:55 WARN Utils: Service 'SparkUI' could not bind on port 4041. Attempting p
ort 4042.
top 10 words = [[34,"of"],[30,"the"],[19,"be"],[19,"to"],[19,"and"],[15,"will"],[12,"from
"],[12,"I"],[11,"freedom"],[10,"that"]]
MacBook-Pro:eclairjs-nashorn billreed$
```

```
var file = "src/test/resources/dream.txt";
var conf = new SparkConf()
                .setAppName("JavaScript word count")
                .setMaster("local[*]");
var sparkContext = new SparkContext(conf);
var rdd = sparkContext.textFile(file).cache();
var rdd2 = rdd.flatMap(function(sentence) {
    return sentence.split(" ");
});
var rdd3 = rdd2.filter(function(word) {
    return word.trim().length > 0;
});
var rdd4 = rdd3.mapToPair(function(word) {
    return [word, 1];
});
var rdd5 = rdd4.reduceByKey(function(a, b) {
    return a + b;
});
var rdd6 = rdd5.mapToPair(function(tuple) {
    return [tuple[1]+0.0, tuple[0]];
})
var rdd7 = rdd6.sortByKey(false);
print("top 10 words = " + JSON.stringify(rdd7.take
(10)));
sparkContext.stop()
```